Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky a výpočetní techniky

Portál pro podporu technologie přeprav nákladních vlaků

Portal for the Support of Technology of Goods Trains Transportation Process

2015 **Jiří Pyszko**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jiří Pyszko**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Portál pro podporu technologie přeprav nákladních vlaků

Portal for the Support of Technology of Goods Trains Transportation Process

Zásady pro vypracování:

Cílem je vytvoření komplexního informačního systému pro podporu technologie přeprav nákladních vlaků. Každá nově zavedená nákladní souprava je doprovázena sadou dokumentů, které musí být zpracovány. Správa dokumentů bude tedy stěžejním úkolem celého projektu.

1. Vytvoření internetové stránky za pomocí technologie ASP.NET MVC.
2. Vytvoření systému pro správu uživatelů. Uživatelé budou do systému přidávání administrátorem a nebudou mít možnost se sami registrovat. Každému uživateli mohou být přiřazena práva pro práci s různými funkcemi systému.
3. Vytvoření komponenty pro práci s dokumenty. Uživatelé mohou vydávat nové dokumenty nebo číst a potvrzovat ty stávající. Dokumenty jsou omezené svou platností, platnost může být ohraničena již při vydání do určitého data, nebo skončí vydáním nového dokumentu. Je nutno rozlišit revizi dokumentů a novou platnost, resp. zneplatnění dokumentů. Neplatné dokumenty jsou uloženy do archivu.
4. Vytvoření komponenty pro tisk sestav. Mezi tyto sestavy patří například dokument o seznámení se s přepravními podmínkami a dokumenty. Tomuto dokumentu se vyplní hlavička a seznam pracovníků přiřazených pro daný projekt přepravy. Dokument se vytiskne, pracovníci jej podepíší a do systému se nahraje jeho naskenovaná kopie.
5. Student navrhne efektivní vyhledávání mezi dokumenty a jejich zálohu, která bude prováděna vždy jednou ročně. Objem dokumentů by neměl přesáhnout 20 tisíc záznamů ročně.

Seznam doporučené odborné literatury:

[1] Troelsen, Andrew. Pro C# and the .NET 4.5 Framework 6th edition. Washington : Apress, 2012. 1430242337.
[2] Brian Driscoll , Nitin Gupta. Entity Framework 6 Recipes. Washington : Apress, 2013. 978-1-4302-5788-2.
[3] Bishop, Judith. C# 3.0 Design Patterns. Indianapolis : O'Reilly Media, 2007. 978-0-596-52773-0.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Plucar**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *30. dubna 2015*

.....
podpis studenta

Poděkování

Rád bych poděkoval ing. Janu Plucararovi za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

V dnešní době se žádná firma neobejde bez řádného informačního systému. Základem dobře fungující firmy je proces seznámení zaměstnanců s potřebnými informacemi a dokumenty. Právě proces informování zaměstnanců bude řešit tato bakalářská práce. Jde o vytvoření systému, který umožní vydávat dokumenty, se kterými se musí zaměstnanci seznámit a potvrdit jejich přečtení. Tento systém má nahradit stávající způsob, jenž používá firma AWT a.s., který spoléhá na rozesílání dokumentů přes elektronickou poštu. Tento způsob byl prokázán jako neefektivní.

Klíčová slova

informační systém, dokumenty, MVC, ASP.NET

Abstract

Nowadays, no company can do without proper information system. In order for the company to function properly it's employees must be familiar with internal processes and documents. Purpouse of this thesis is to improve document distribution process. It's main goal is to create a system that allows document publishing. These documents must be red and confirmed by users. This system should replace currently used process in AWT a.s. company. Nowadays documents are sent via email, which is not effective enough.

Key words

information system, documents, MVC, ASP.NET

Seznam použitých zkratek

Zkratka	Význam
ASP.NET	Active Server Pages .NET
IS	Informační systém
MVC	Model, View, Controler
RUP	Rational Unified Process
LINQ	Language Integrated Query
FURPS	Functionality, Usability, Reliability, Performance, Supportability
SQL	Structured Query Language
API	Aplication Programming Interface
CSS	Cascading Style Sheet
HTML	Hyper Text Markup Language
PDF	Portable Document Format
SCM	Software Configuration Management

Obsah

Úvod

Tato práce se zabývá problematikou informování zaměstnanců dané firmy, o tom, kdy jsou vydávány nové dokumenty, které musí určití zaměstnanci přečíst. Systém by měl nabízet jednoduchý a přehledný způsob, jak být informován o nově vydaných dokumentech.

Práce je rozdělena na několik částí. První část projektu je věnována analýze, která řeší požadavky uživatelů, firemní procesy a případy užití. V další kapitole je popsán návrh architektury systému. Návrh bude vytvořen na základě požadavků, které jsou na něj kladeny ve fázi analýzy. Druhá polovina práce se zaměřuje na implementaci systému, jednotlivá řešení různých technických problémů a použité technologie, které byly využívány při vývoji.

Výsledným produktem bude systém, který zpřehlední práci s dokumenty a předpisy, které zaměstnanci potřebují a kontrola, zda si dané dokumenty přečetli a potvrdili.

1 Popis systému

Tato bakalářská práce se věnuje vývoji informačního systému, který bude uchovávat dokumenty a předpisy pro zaměstnance. Zaměstnanec, pro kterého bude daný předpis určen, se bude muset přihlásit do systému pomocí svého účtu. Uživatelé se do systému nemohou sami registrovat. Administrátor systému vytvoří pro uživatele účet během procesu zaškolení. Po přihlášení se uživatel dostane na stránku s přiřazenými dokumenty. Na této stránce bude mít uživatel možnost stáhnout daný dokument, přečíst si ho a potvrdit jeho přečtení. Dokumenty budou obsahovat informaci o datu, do kterého musí zaměstnanci potvrdit, že jej přečetli. Údaje o potvrzení seznámení se s dokumentem se uloží do databáze, aby bylo možné zjistit, kteří uživatelé potvrdili přečtení, kteří jej ještě nepotvrdili, nebo ty, kteří dokumenty potvrdit nestihli. Ke každému dokumentu je možnost vytvořit tiskovou sestavu se jmény zaměstnanců. Tu jednotliví zaměstnanci podepíší po seznámení se s dokumentem. Tento list s podpisy bude možné naskenovat a nahrát do systému.

Informační systém bude v provozu neustále a bude dostupný prostřednictvím webového rozhraní, ke kterému se zaměstnanci budou moci připojit odkudkoli. Informace o uživatelích a dokumentech budou uloženy v databázi a samotné soubory budou uloženy v datovém úložišti.

2 Postup vytvoření systému

V rámci vytváření softwarových systémů si je možné vybrat z mnoha různorodých způsobů návrhů a postupů, které povedou k realizaci cílového systému. Pro tento systém byla zvolena metoda Rational Unified Process (RUP).

RUP je softwarový inženýrský proces. Poskytuje disciplinovaný přístup k přidělování úkolů a povinností pro vývoj. Jeho cílem je zajistit produkci vysoce kvalitního softwaru, který splňuje všechny požadavky cílového uživatele v předvídatelném harmonogramu a rozpočtu. RUP je vyvíjeno a zpravováno firmou Rational Software [1].

Podle [1] můžeme vývoj systému rozdělit z časového hlediska na 4 fáze. Každá fáze je zakončena milníkem - bodem v čase, který obsahuje kritická rozhodnutí, která se musí provést, aby byly dosaženy hlavní cíle vývoje.

Zahajovací fáze (Inception)

Během této fáze se vývojář soustředí na začátek vývoje. Získávají se požadavky na systém, domlouvá se, co je možné realizovat a odhaduje se cena systému. Důležité je zjistit, které subjekty budou se systémem pracovat a identifikují se klíčové případy užití.

Cílem této fáze by pak měl být dokument s vizí systému, základní model případu užití, základní zhodnocení rizik, projektový plán, zobrazení fází a iterací.

Fáze rozpracování (Elaboration)

Cílem této fáze je analyzovat problémy, vytvořit si projektový plán, posoudit rizika a určit si architekturu systému, který bude vytvořen. Jsou prováděny klíčová architektonická rozhodnutí, která se vztahují na celý systém.

Výsledkem této fáze je spustitelný prototyp. Měly by být z větší části hotové případy užití, popis architektury, analýza rizik a vytvořen plán vývoje celého projektu.

Konstrukční fáze (Construction)

V konstrukční fázi dochází k implementaci informačního systému, kdy jsou všechny komponenty vyvíjeny a postupně integrovány do systému. Veškerý vytvořený kód je při tom testován.

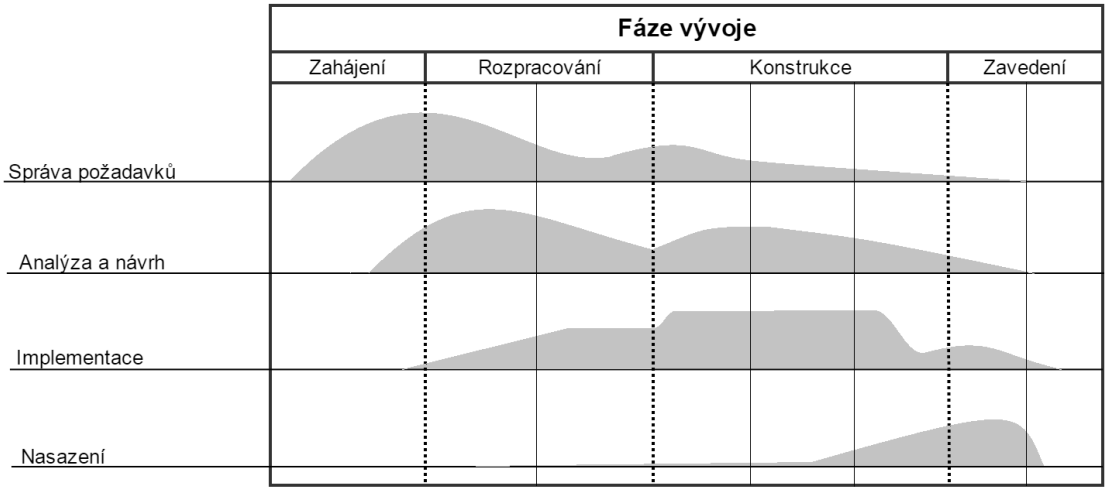
Na konci této fáze by měl být vytvořen funkční systém. Dále je vypracován uživatelský manuál, který má pomoci uživatelům s orientací v systému a je vypracován popis aktuální verze sytému.

Fáze zavedení (Transition)

Poslední fáze se zabývá přenesením softwaru od vývojáře do uživatelského provozu. Předáním softwaru cílovému subjektu vývoj systému nekončí. Je potřeba zajistit opravu problémů, které vznikly při nasazování systému. Obvykle jsou vydány nové verze systému,

ve kterých jsou opraveny problémy z předešlé verze. Konec zavedení nastává tehdy, když je systém úspěšně nahrán a spuštěn na uživatelské doméně.

Při zavádění nového systému se často používá po určitou dobu i starý systém a oba jsou provozovány paralelně. Je to ochrana proti případnému pádu nového systému. Musíme zajistit první testování uživateli - beta testing, který zkontroluje správnost všech funkcí v systému. Je třeba počítat i se zaškolením pracovníků.



Obrázek 1.1: *RUP Fáze vývoje*

Na obrázku 1.1 je znázorněn pohled na RUP fáze, kdy na horizontální ose jsou zobrazeny jednotlivé fáze a iterace vývoje systému a na svislé ose jsou zobrazeny reprezentace aspektů procesu. Každá iterace trvá přibližně 1-2 týdny, kdy po každé iteraci byla provedena konzultace.

Z obrázku 1.1 lze pozorovat postup při vývoji celého systému. Nejprve v první fázi byly zjištěny požadavky, které jsou na systém kladeny a částečně analyzovány. V druhé fázi byla provedena hlavní část analýzy, začala se vytvářet implementace systému a klesaly nároky na správu požadavku. Ve třetí fázi je pak provedena většina implementace. Ve fázi zavedení se pak věnuje čas hlavně nasazení systému na cílové servery.

3 Analýza systému

Jak již bylo v minulé kapitole naznačeno, vývoj softwaru neznamená jenom psaní kódu. Je potřeba mít představu o vytvářeném softwaru. Toho dosáhneme analýzou zadání. Ta obsahuje popis vyvíjeného systému, jak má pracovat a jak bude ve výsledku vypadat. Právě tomu se věnuje kapitola analýzy. Nejjednodušším způsobem, jak získat základní představu o systému je použití jednoduchých otázek. Tento způsob analýzy řeší Zachmanův Framework.

3.1 Zachmanův Framework

Zachmanův Framework je základní architektonický framework, který poskytuje formální a vysoce strukturovaný způsob zobrazení a definování systému. Skládá se z dvojrozměrné matice na základě šesti základních otázek: co, kde, kdy, proč, kdo a jak s pěti úrovněmi zhmotnění. Postupně transformuje abstraktní myšlenky až na úroveň konkrétních nápadů[10].

Tato práce využívá jen nejvyšší vrstvu Zachmanova Frameworku, která sleduje systém z pohledu projektanta. Popisuje modely, architekturu a reprezentace, které zobrazují hranice systému a vysvětlují, které se musí brát v úvahu.

Na obrázku 1.2 je znázorněna tabulka Zachmanova Frameworku, která je obvykle znázorněna jako matice 6 x 6 s poli pro popis. Na obrázku lze vidět, že systém se zabývá nejvyšší vrstvou Zachmanova Frameworku, kdy se bere v potaz systém z pohledu projektanta.

	Data	Funkce	Síť	Lidé	Čas	Motivace
	Co	Jak	Kde	Kdo	Kdy	Proč
Cíl/Rozsah Role: Projektant	Seznam důležitých věcí pro firmu	Seznam firemních procesů	Seznam firemních míst	Seznam důležitých organizací	Seznam událostí	Seznam firemních cílů a strategií
Podnikový model Role: Vlastník	Konceptuální datový model	Firemní model procesů	Firemní logický model	Pracovní postup	Časový plán	Firemní plán
Systémový model Role: Návrhář	Logický datový model	Model systémové architektury	Architektura distribuovaných systému	Architektura rozhraní mezi člověkem a systémem	Struktura procesů	Model firemních pravidel
Technologický model Role: Architekt	Fyzický Datový Model	Technologický návrh	Architektura technologie	Prezenční architektura	Struktura řízení	Návrh pravidel
Podrobné zastoupení Role: Programátor	Definice dat	Program	Architektura sítě	Architektura bezpečnosti	Definice časování	Specifikace pravidel
Funkční podnik Role: Uživatel	Použitelná data	Fungující funkce	Použitelná síť	Fungující organizace	Implemento-vaný časový rozvrh	Strategie práce

Obrázek 1.2: *Zobrazení Zachmanova frameworku*

3.1.1 Otázky Zachmanova frameworku

Co?

Webový portál, který bude zajišťovat firemní správu dokumentů a práci s nimi.

Jak?

Systém bude zvládat:

- přístup k dokumentům
- editaci dokumentů
- zápis zaměstnance, který se s dokumentem seznámil
- kontrolu platnosti dokumentů
- automatické upozornění na nutnost seznámeni s dokumentem

Kde?

Systém bude pracovat na serveru pomocí webového rozhraní a bude k němu mít přístup každý s uživatelským účtem.

Kdo?

Systém je určen pro zaměstnance dané firmy.

Kdy?

- Systém bude spuštěn v nepřetržitém provozu.
- Zaměstnanci budou muset reagovat na dokument jen tehdy, kdy je pro ně přímo určen a mají si ho přečíst a označit jako přečtený.
- Zaměstnanci budou automaticky upozorněni na nutnost přečtení dokumentu.

Proč?

Portál bude sloužit ke zpřehlednění práci s předpisy a dokumenty a pro kontrolu, zda jsou zaměstnanci seznámeni s danými dokumenty.

3.2 FURPS

FURPS model byl původně vyvinut Robertem Gradym z firmy Hewlett Packard a později rozšířen firmou Rational Software, nyní IBM Rational Software. Tento model na systém pohlíží z pěti úhlů. Je vytvořen seznam různých požadavků na systém a každý požadavek má své označení, název, popis a prioritu, kterou v systému zastupuje.

3.2.1 Funkčnost

Funkčnost představuje funkční požadavky na sytém. Toto obvykle zobrazuje hlavní znaky systému, jeho hlavní funkčnosti a schopnosti.

Označení: FP1

Název: Vkládání nových dokumentů

Popis: Systém zvládne zpracovat dokument a uložit ho do databáze

Priorita: 10

Označení: FP2

Název: Oznámení o novém důležitém dokumentu

Popis: Systém pošle email uživateli, kterému je dokument určen, aby ho upozornil na potřebu zkontrolovat nově přiřazené dokumenty

Priorita: 7

Označení: FP3

Název: Potvrzení o přečtení dokumentu

Popis: Systém zaznamená, že uživatel potvrdil přečtení dokumentu

Priorita: 10

Označení: FP4

Název: Přihlášení do systému

Popis: V systému budou vytvořeny účty uživatelům a možnost přihlásit se přes svůj účet

Priorita: 8

Označení: FP5

Název: Tisk podpisových formulářů

Popis: Systém dokáže vytvořit tiskovou sestavu se jmény zaměstnanců přiřazených k dokumentu, aby bylo možno ho podepsat i v tištěné podobě

Priorita: 7

3.2.2 Vhodnost k použití

Do kategorie vhodnost k použití jsou zařazeny požadavky na systém z pohledu používání systému

Označení: VP 1

Název: Intuitivní ovládací prvky

Popis: Vytvoření grafického rozhraní, které zvládnou ovládat i nezkušení uživatelé

Priorita: 8

Označení: VP 2

Název: Návod na použití

Popis: Vypracování návodu, aby systém mohli používat i nezkušení uživatelé

Priorita: 4

3.2.3 **Spolehlivost**

Spolehlivost zahrnuje požadavky, jako jsou dostupnost, přesnost, využitelnost a odolnost vůči chybám.

Označení: SP 1

Název: Odolnost vůči chybám

Popis: Systém musí být odolný vůči uživatelským chybám, které vzniknou při obsluze systému, validace formulářů, datových typů

Priorita: 9

Označení: SP 2

Název: Znovuspuštění po pádu

Popis: Služby systému se automaticky restartují po spuštění serveru

Priorita: 6

Označení: SP 3

Název: Systém bude v provozu 24h/ denně

Popis: Systém bude přístupný v jakoukoli denní dobu

Priorita: 7

3.2.4 Schopnost být udržován

Sleduje systém z hlediska udržování aplikace. Obsahuje řadu požadavků jako je snadné testování, přizpůsobivost, udržovatelnost, kompatibilita nebo nasazení.

Označení: UP 1

Název: Nasazení systému

Popis: Systém se nasadí do provozu, předá se dokumentace, provede se zaškolení zaměstnanců

Priorita: 9

Označení: UP 2

Název: Možnost rozšíření

Popis: Pokud bude potřeba, bude možnost rozšířit systém na větší zátěž, než na kterou je systém nastaven a to použitím balančního serveru

Priorita: 4

3.2.5 Výkon

Výkon zahrnuje propustnost systému, rychlost odezvy systému, dobu zotavení systému, spuštěný čas a všechny ostatní aspekty, které určují, jak výkonný musí být systém a hardware, aby byl systém vhodný k použití.

Označení: UP 1

Název: Zpracování více uživatelů

Popis: Systém musí být schopný obsluhovat paralelně až 200 uživatelů

Priorita: 7

Označení: UP 2

Název: Uchovávání dokumentů 5 let

Popis: Systém musí zvládat uchovávat dokumenty 5 let do minulosti

Priorita: 7

Označení: UP 3

Název: Rychlé načtení stránky

Popis: Systém načte stránku do 2 sekund

Priorita: 7

3.3 Scénáře případů užití (Use Case scenarios)

Scénáře případů užití znázorňuje interakce mezi uživatelem a systémem. Popisují krok za krokem, jak se uživatel dostane ke specifickému cíli. Navíc jsou zde zaznamenány i případy, kdy se v systému objeví něco špatně a možnosti řešení těchto problémů.

Identifikátor: Označení případu užití

Název: Název případu užití

Vstupní podmínky: Obsahuje stav systému před spuštěním případu užití

Úspěšný cílový stav: Obsahuje stav, v jaké se systém nachází po spuštění případu užití

Aktér: Uživatelé, kteří budou s případem užití pracovat

Hlavní scénář: Popisuje ideální průběh interakce uživatele se systémem za účelem dosažení cíle

Výjimky: Popisuje události, které se provedou, pokud dojde k odchylce od hlavního scénáře

Alternativy: Odchylky od hlavního scénáře, které povedou k úspěšnému dokončení hlavního scénáře

Identifikátor: UC7

Název: Přidání dokumentu

Vstupní podmínky: Uživatel musí být přihlášen

Úspěšný cílový stav: Dokument je v systému uložen a je odesláno upozornění pro zvolené zaměstnance

Aktér: Manažer, Administrátor

Hlavní scénář:

1. Uživatel zvolí možnost nahrát nový dokument
2. Systém zobrazí formulář pro vkládání nového dokumentu
3. Uživatel stiskne ovládací prvek pro výběr souboru
4. Systém zobrazí soubory v lokálním PC
5. Uživatel vybere soubor
6. Systém potvrdí vybraný soubor
7. Uživatel vyplní údaje vztahující se k dokumentu a potvrdí vložení
8. Systém uloží údaje a soubor do databáze a uvědomí zaměstnance

Výjimky:

7a. Uživatel vyplnil špatný formát dat

7b. Systém znovu zobrazí formulář na přidání dokumentu

8a. Systému se nepodařilo uložit dokument do databáze z důvodu nepřístupnosti databáze

8b. Uživateli je zobrazena zpráva „Nepodařilo se uložit dokument, zkuste to prosím později“ a nabídnuta možnost zkusit znovu a zkusit později

Alternativy:

3. Uživatel vybere soubor pomocí odkazu

Identifikátor: UC5

Název: Smazání dokumentu

Vstupní podmínky: Uživatel musí být přihlášen a mít právo na úpravu souboru

Úspěšný cílový stav: Dokument je odstraněn z databáze

Aktér: Manažer, Administrátor

Hlavní scénář:

1. Uživatel přejde na stránku s dokumenty
2. Systém zobrazí stránku s dokumenty
3. Uživatel zadá jméno do vyhledávacího okna
4. Systém zobrazí jen dokumenty, které uživatel vyhledává
5. Uživatel vybere soubor a stiskne smazat
6. Systém zobrazí potvrzovací formulář
7. Uživatel potvrdí smazání
8. Systém smaže dokument a uživatel je přesměrován zpět na stránku s dokumenty

Výjimky:

3. Uživatel nemá pravomoci na smazání dokumentu

3a. Uživatel použije účet, který má právo pro smazání daného dokumentu

6. Systému se nepodařilo smazat dokument do databáze z důvodu nepřístupnosti databáze

6b. Uživateli je zobrazena zpráva „Nepodařilo se smazat dokument “ a nabídnuta možnost zkusit znovu nebo zkusit později

Alternativy

5. Uživatel zvolil, že nechce daný dokument smazat a je mu znovu zobrazen formulář vyhledaných dokumentů

Identifikátor: UC2

Název: Vyhledávání dokumentů

Vstupní podmínky: Uživatel musí být přihlášen a zvolit vyhledávání

Úspěšný cílový stav: Uživateli se zobrazí seznam vyhledaných dokumentů

Aktér: Uživatel, Manažer, Administrátor

Hlavní scénář:

1. Uživatel vstoupí na stránku s dokumenty
2. Systém zobrazí všechny dokumenty
3. Uživatel vyplní údaje v kolonce vyhledávání a vybere možnost vyhledat
4. Systém zobrazí jen dokumenty, s požadovanými údaji
5. ext Uc3

Výjimky:

4. Systém nenalezl žádné dokumenty souhlasící s požadavky uživatele

4a. Systém zobrazí zprávu o nenalezením dokumentu

Alternativy:

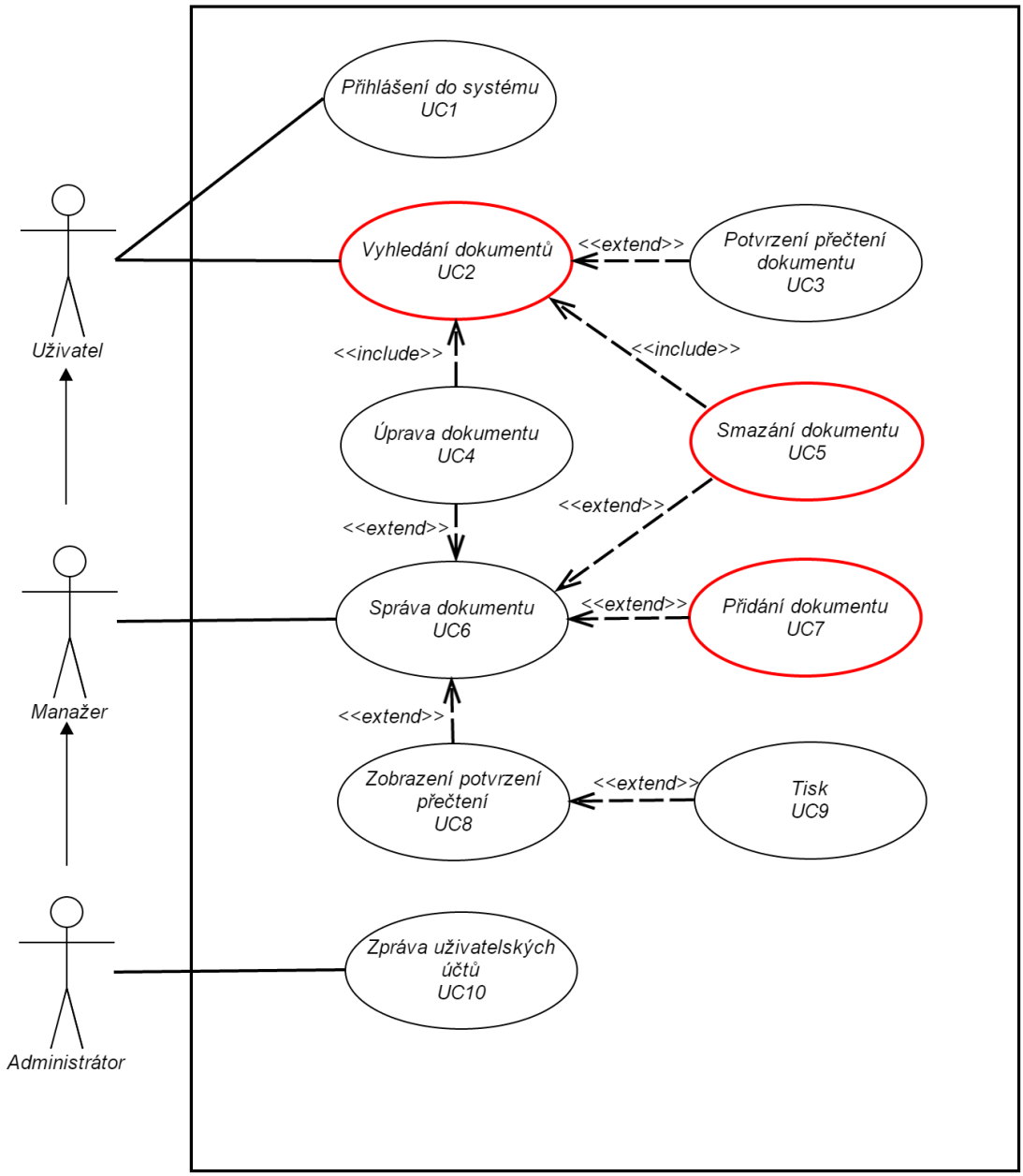
3a. Uživatel najde a vybere soubor ze seznamu všech dokumentů pouhým procházením všech dokumentů

3.4 Náhled na scénáře případu užití

Pro vizualizaci scénářů a interakcí mezi nimi je použit Diagram případu užití (Use case diagram).

Diagram případu užití se používá k popisu chování systému z hlediska uživatele a zachycuje, které typy uživatelů se systémem pracují a jaké činnosti v rámci systému vykonávají. Umožňuje znázornit funkční požadavky na systém tím, že popisuje interakci mezi systémem a uživateli.

Obrázek 1.3 popisuje tento systém právě z hlediska diagramu případu užití. Systém počítá s vytvořením tří typů uživatelských účtů. Uživatelský, který má přístup jen k základnímu náhledu dokumentů a potvrzení přečtení dokumentů. Manažerský, který dovoluje práci s jednotlivými dokumenty a administrátorský, který umožňuje zpravovat jednotlivé uživatelské účty.

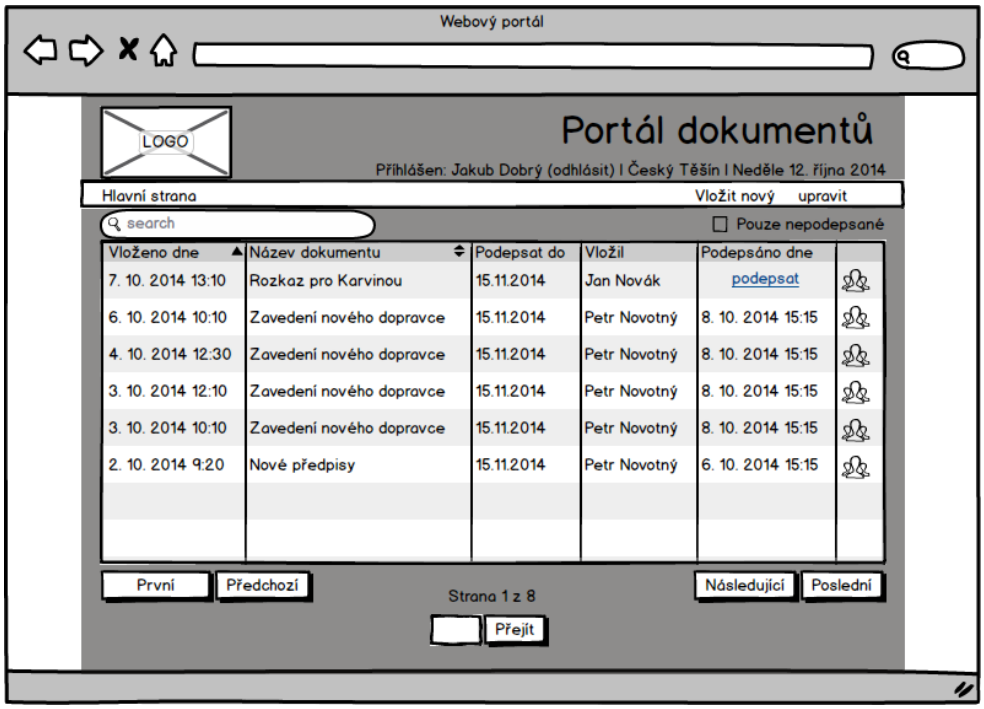


Obrázek 1.3: *Diagram případu užití systému*

3.5 Nákresy systému

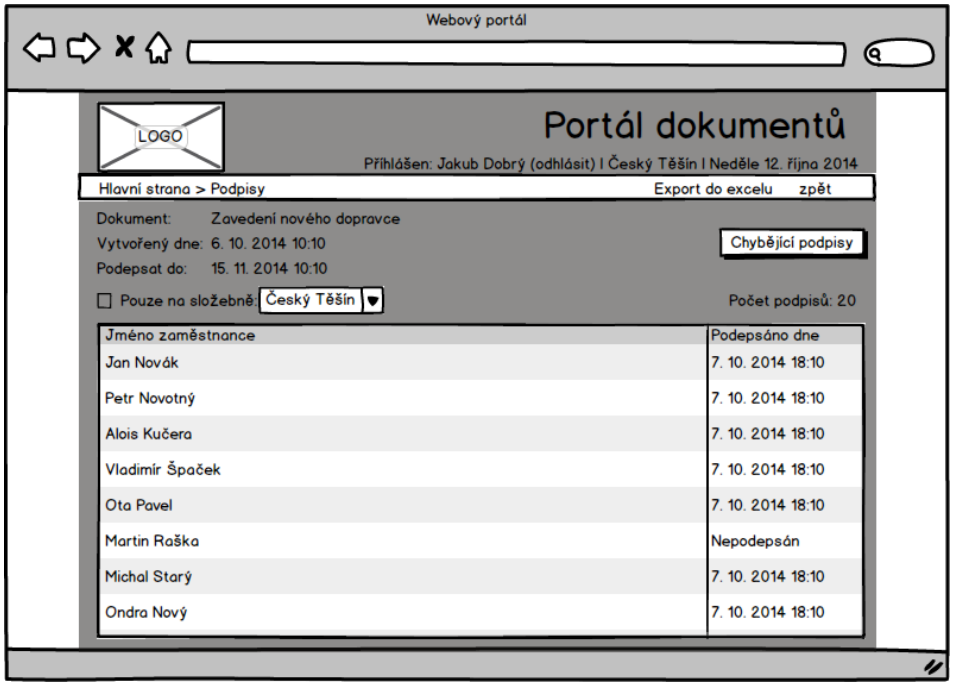
Nákres je ilustrace grafického rozhraní, která se zaměřuje na rozvržení ovládacích prvků, priorita obsahu, rozvržení místa a předpokládané chování. Díky těmto důvodům nákresy systému obvykle neobsahují žádné stylování ani barvy. Nákresy navíc pomáhají zobrazit vztahy mezi jednotlivými stránkami.

Na obrázku 1.4 je zobrazen nákres hlavní strany, na kterou uživatel přejde po přihlášení do systému. Zde nalezne potřebné ovládací prvky a bude mít přístup k veškerým dokumentům, které jsou pro daného uživatele určeny. Administrátor bude mít navíc k dispozici ovládací prvky pro zprávu uživatelských účtů.



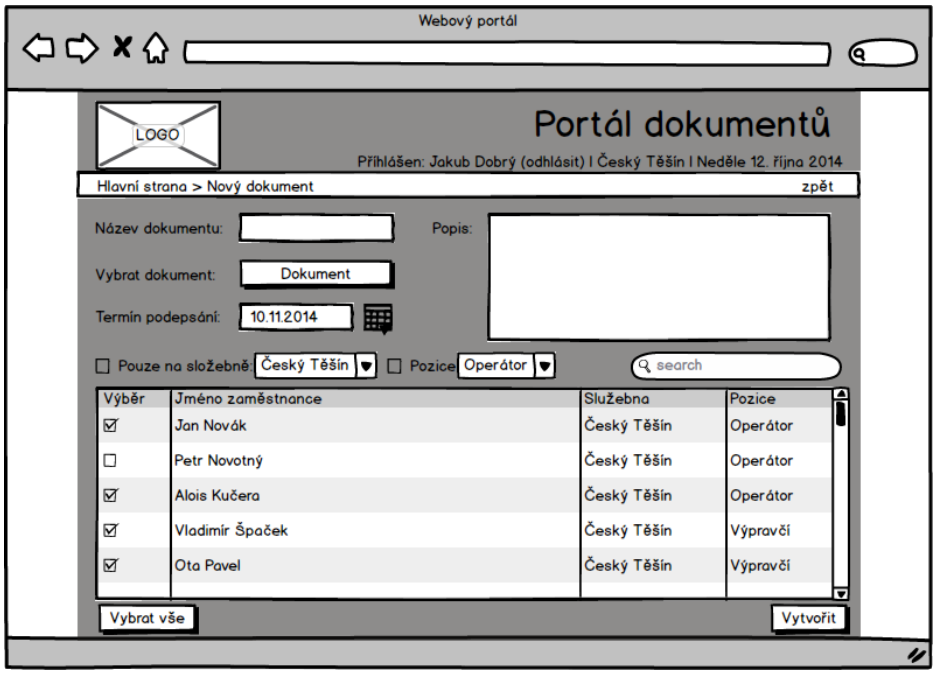
Obrázek 1.4: *Nákres hlavní strany*

Na obrázku 1.5 je znázorněn list potvrzení o přečtení, který s daným dokumentem souvisí. Tato stránka bude využita k vytvoření tiskových zpráv zobrazujících, zda byl dokument potvrzen všemi uživateli, nebo někdo dokument potvrdil.



Obrázek 1.5: *Nákres zobrazení podpisů*

Na obrázku 1.6 je znázorněna ukázka předpokládaného vzhledu stránky pro přidávání nových dokumentů, ke které nebudou mít přístup uživatelé systému, ale jen uživatelé s dostatečnými právy pro manipulaci s databází dokumentů. Zde se vloží dokument a přiřadí se k jednotlivým zaměstnancům. Po potvrzení přečtení se informace o dokumentu uloží do databáze.



Obrázek 1.6: *Nákres vložení nového dokumentu*

3.6 Technická specifikace

Technická specifikace pro úložiště počítá s počtem uživatelů kolem 2 000 při počtu nových dokumentů přibližně 20 000 ročně. Dokumenty budou ve formátu portable document format (PDF) a přibližná velikost souboru bude 0.5 – 5 MB. Velikost prostoru na disku by tedy neměla přesáhnout více, než 100 GB dat v dokumentech za rok.

Dalším důležitým prvkem systému je jeho rychlost. V tomto případě hrají největší roli prvky jako je procesor nebo paměti. Jelikož se předpokládá, že počet uživatelů bude asi jen kolem 2 000, tak se na výkon těchto dvou prvků neklade velký důraz.

Systém používá databázi, která je v tomto případě MS SQL. Server musí podporovat právě tento typ databáze.

Dalším požadavkem je, že server musí zvládat spouštět aplikace typu ASP.NET MVC. Pokud bude splňovat jen podmínku ASP.NET, tak prezentační vrstva aplikace nebude funkční a systém nebude fungovat.

Způsoby řešení serverů

Lokální server, kdy má cílový uživatel vlastní server s dostatečným výkonem a splňuje všechny požadavky, pro spuštění a běh informačního systému. Nevýhodou je, že o všechny zálohy systému a o provoz serveru se někdo musí starat.

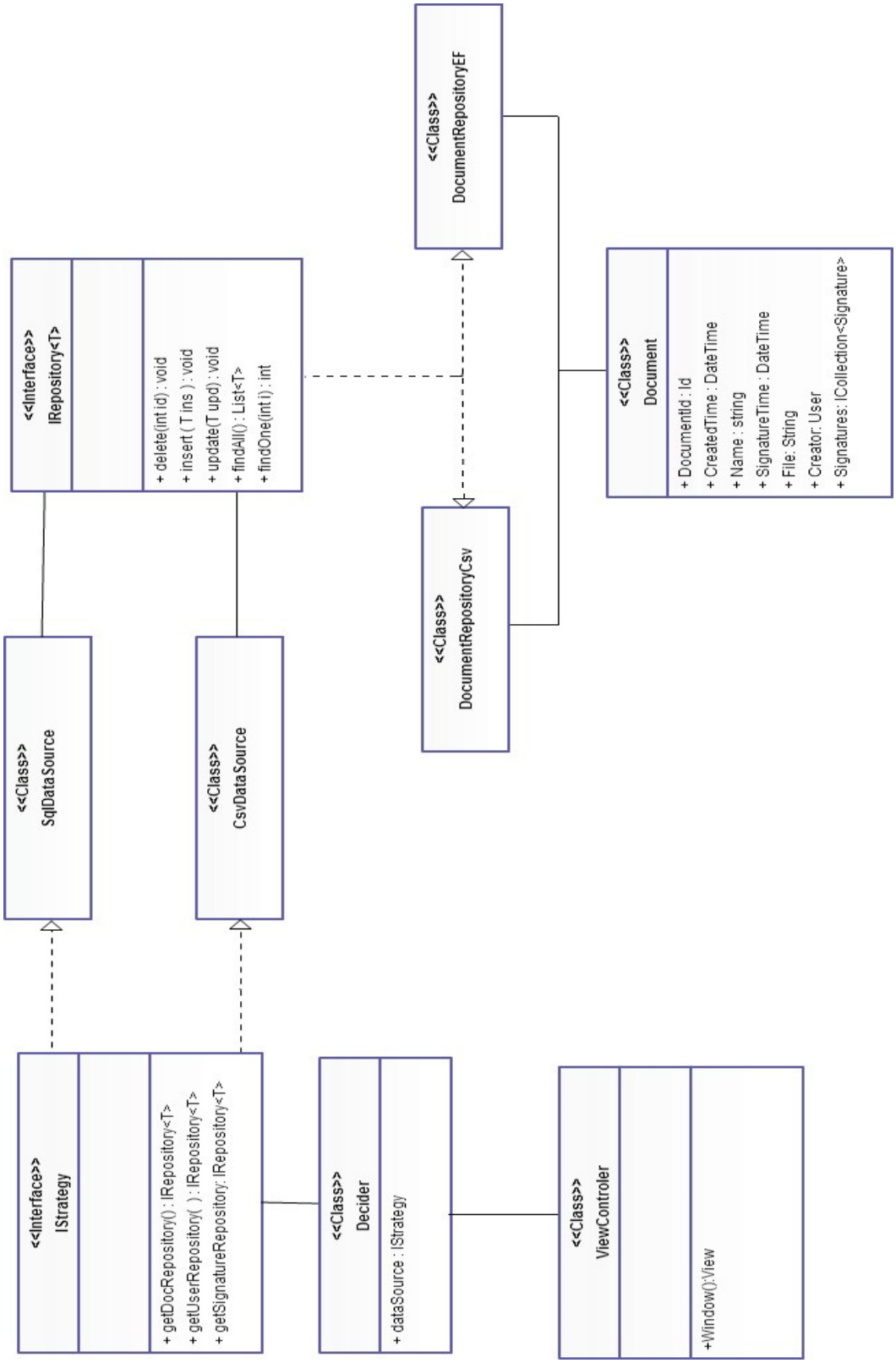
Pronájem virtuálního serveru s doménou je jednoduchá možnost, jak provozovat tento informační systém, aniž by měl daný subjekt vlastní server. Nevýhodou jsem poplatky za vedení účtu a domény.

3.7 Vnitřní struktura systému

Vnitřní struktura systému je vizualizována použitím třídního diagramu.

Třídní diagram je statický pohled na aplikaci. Třídní diagram není použit jen na vizualizaci, popis a dokumentaci různých částí systému, ale také pro vytvoření spustitelného kódu aplikace. Popisuje atributy a operace tříd. Třídní diagramy jsou široce používané pro sestavení objektově orientovaných systémů, protože jsou to jediné UML diagramy, které můžou být mapovaný přímo požadovaným jazykem.

Na obrázku 1.7 je zobrazen návrh vnitřní struktury systému, který ukazuje strukturu systému pro práci s dokumenty. Uživatel ovládá program pomocí grafického rozhraní, a interakce z něho zpracovává controler, v tomto případě ViewController. ViewController pošle informace do třídy Decider, která na základě strategie rozhoduje, jaká technologie pro přenos dat a úložiště se použije.



Obrázek 1.7: *Vnitřní struktura systému*

4 Architektura systému

Tato sekce je věnována prostředkům, technologiím a vzorům, které jsem použil pro vytváření výsledné aplikace.

4.1 Vícevrstvá architektura

Vícevrstvá architektura rozděluje systém na více logicky oddělených částí. Takto vytvořená aplikace se dá taky nazvat jako distribuovaná aplikace nebo vícevrstvá aplikace. Typickým příkladem vícevrstvé architektury je třívrstvá architektura. Ta rozděluje systém na 3 separátní části: prezentační vrstva, střední vrstva a datová vrstva. Nejjednodušším způsobem, jak rozdělit aplikaci na vícevrstvou je vytvořit pro každou vrstvu vlastní projekt a všechny projekty vložit do jednoho řešení.

4.1.1 Prezentační vrstva

Prezentační vrstva je vrstva, se kterou přichází do styku uživatel dané aplikace. Obvykle obsahuje informace, které chtěl uživatel získat, kontrolní prvky, které vysílají uživatelské požadavky. Prezentační vrstva obvykle přistupuje k prostřední vrstvě pomocí referencí. Prezentační vrstva přímo nepřistupuje k datové vrstvě, místo toho komunikuje s datovou vrstvou pomocí střední vrstvy.

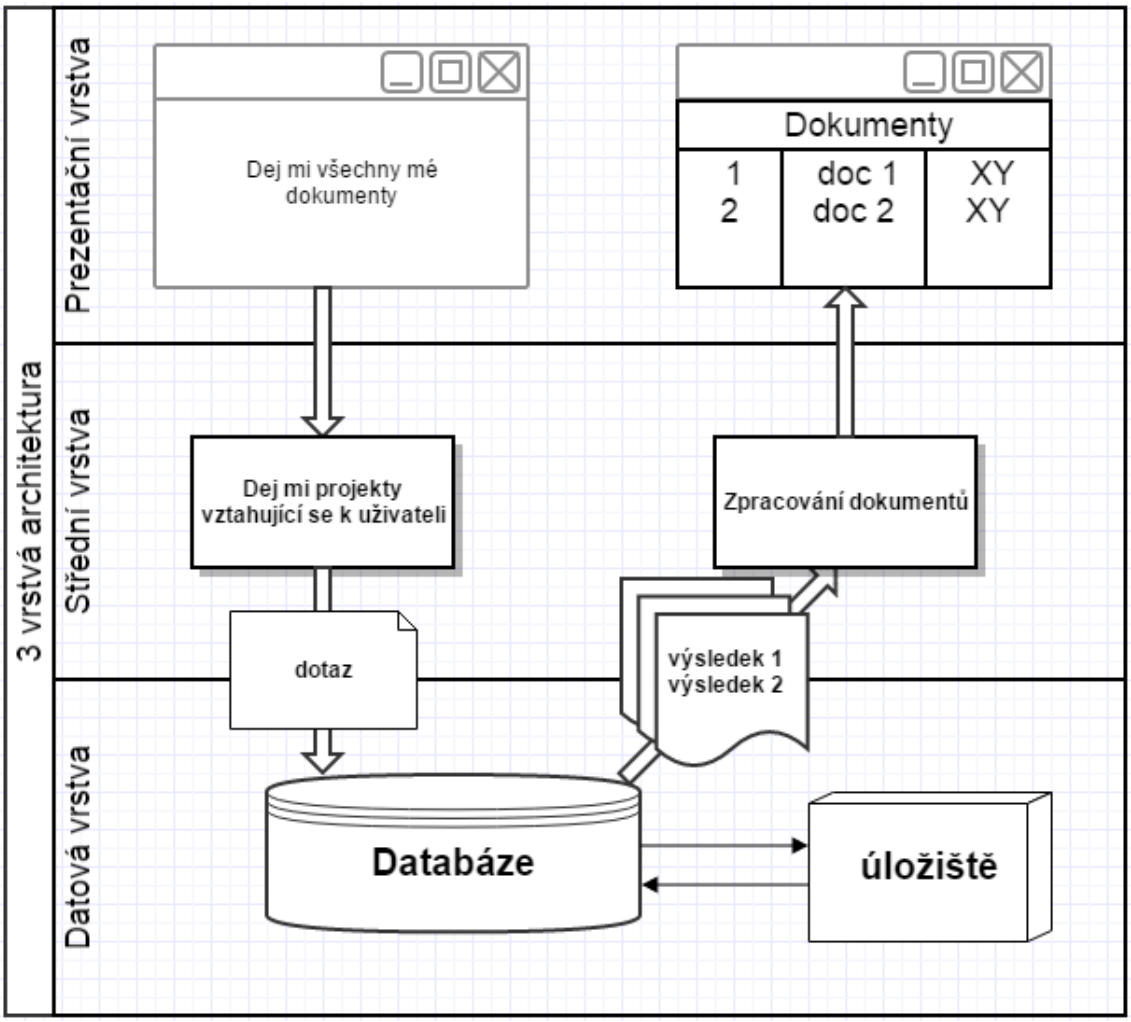
4.1.2 Střední vrstva

Střední vrstva nebo také byznys vrstva je vrstva, která slouží jako prostředník mezi datovou vrstvou a prezentační vrstvou. Poskytuje komponenty, jako jsou validace dat, správa přihlašování, objektová reprezentace dat a mnoho dalších.

4.1.3 Datová vrstva

Datová vrstva ukládá aplikační data a nabízí přístup k nim. Datová vrstva není přímo přístupná z prezentační vrstvy, ale musí se na ní přistupovat z vrstvy střední.

Na obrázku 1.8 je zobrazeno, jak probíhá komunikace v třívrstvé architektuře. Uživatel má přístup jenom k prezentační vrstvě. Uživatelské rozhraní zavolá metodu, jenž vrací dokumenty uživatele. Tento příkaz zpracovává střední vrstva, která ho předá datové vrstvě, která ho převede na SQL dotaz. V datové vrstvě se dotaz provede a hodnoty z databáze se vrátí zpět do střední vrstvy. Střední vrstva poté zpracuje data, popřípadě je upraví a přepošle je na stránku z prezentační vrstvy.



Obrázek 1.8: *Třívrstvá architektura*

4.2 Použité návrhové vzory

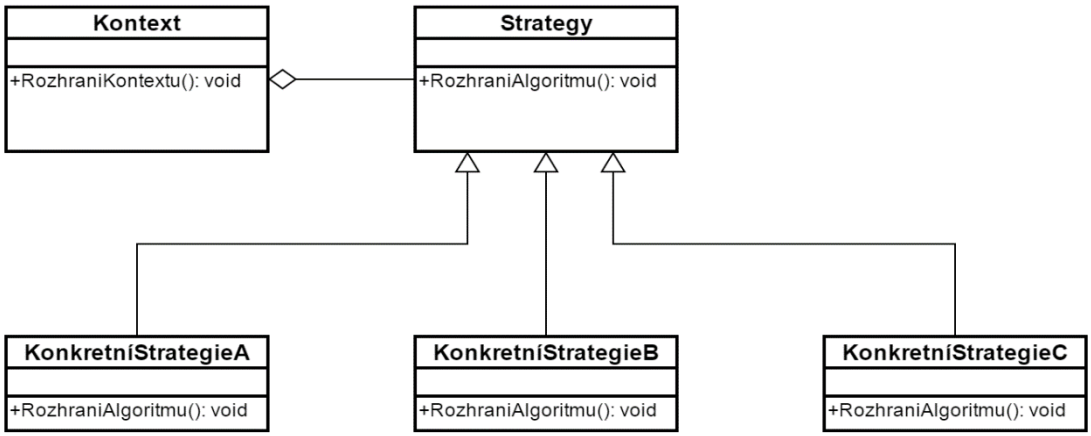
V softwarovém vývoji se myslí návrhovým vzorem abstraktní popis řešení problému, který se vyskytuje v mnoha projektech. Vývojáři pak použijí tento vzor, aby vyřešili problém ve svých specifických projektech.

4.2.1 Návrhový vzor strategie

Návrhový vzor strategie (Strategy pattern) slouží k vyměňování různých implementací algoritmu za běhu programu. Tato záměna může proběhnout buď explicitně (na žádost klienta) nebo implicitně (na základě nastavení ovladače)[6].

V tomto systému je návrhový vzor použit na zvolení zdroje dat. Implicitně se data získávají z databáze, ale pomocí návrhového zdroje strategie je možnost změnit zdroj dat jen upravením jedné hodnoty. To nám umožňuje vyměnit dané SŘBD pomocí jednoho příkazu. Zdroj dat se pak může změnit z databáze na zdroj dat ze souboru.

Na obrázku 1.9 je příklad obecného návrhového vzoru strategie. Kontext zde slouží jako výběrčí, ve kterém se vybírá, která strategie se použije. Všechny 3 strategie (A, B, C) obsahují funkci, která má stejnou deklaraci, ale definice se v každé strategii liší. Tím lze zajisti to, že jednoduchou změnou lze změnit způsob vykonání funkce. V tomto systému lze tímto změnit zdroj dat z databáze na jiný zdroj dat.



Obrázek 1.9: *Návrhový vzor strategie*

4.2.2 Návrhový vzor repozitář

V mnoha aplikacích se stává, že doménová logika je napojena přímo na úložiště dat, jako jsou databáze nebo webové služby. Toto ale může vyústit v mnoha problémech jako je duplicita dat, vyššímu výskytu chyb nebo nemožností testovat byznys logiku v izolovaném prostředí. Použitím návrhového vzoru repozitář můžeme těmto problémům předejít. Repozitář dokáže oddělit logiku, která získává data a mapuje je na entitní model z byznys logiky, která reaguje na model[7].

Repozitář zprostředkovává vazbu mezi datovou vrstvou a střední vrstvou aplikace. Dotazuje se na data ze zdroje dat, mapuje data do entit a mění data ve zdroji podle pokynů z logiky střední vrstvy.

V systému je návrhový vzor repozitář implementován za účelem práce s daty která se získávají z databáze a zase se vkládají zpět.

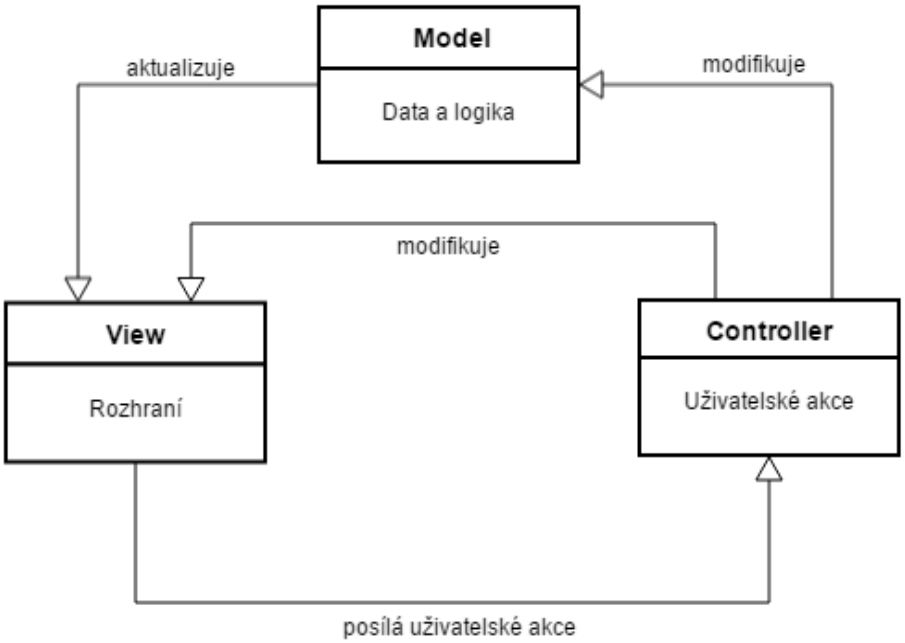
Na obrázku 1.10 je znázorněno schéma repozitáře, kde si klient pomocí klientské doménové logiky požádá o informace. Repozitář příkaz zpracuje a pomocí mapování dat převede data ze zdroje dat na objekty. Klientská logika pak získá výstup dat z repozitáře.

Obrázek 1.10: *Návrhový vzor repozitář*

4.3 MVC

Model-View-Controller je architektonický model, který rozděluje aplikaci do 3 komponent: model, view a controller. Pomáhá vytvořit aplikaci, která odděluje její různé aspekty, přičemž poskytuje a zpracovává vztahy mezi jednotlivými komponentami systému. Návrhový vzor specifikuje, kde se jednotlivé prvky mají nacházet v aplikaci[2].

Na obrázku 1.11 je znázorněno, jak spolu všechny tři komponenty komunikují. View posílá akce do contolleru, ten je zpracovává, a podle toho může modifikovat view nebo model. Pokud controller upraví data v modelu, tak se při znovunačtení dat zobrazí změněná hodnota z modelu. Hlavní je to, že samotný view nemá přístup přímo do modelu, že by ho dokázal upravit, ale vše je řízeno controllerem, ve kterém by měla být uložena většina logiky systému.



Obrázek 1.11: *Nákres MVC*

Model - Modelové objekty jsou tou částí aplikace, která implementuje logiku pro aplikační datovou doménu. Často je modelový objekt získáván a ukládán do databáze. Reprezentuje informace, se kterými se pracuje.

View - View je komponenta, která poskytuje aplikaci uživatelské rozhraní. Poskytuje uživateli ovládací prvky, pomocí kterých uživatel s programem komunikuje. Do této komponenty patří veškeré grafické zobrazení.

Controller - Controller je komponenta, která zpracovává uživatelské akce, pracuje s modelem a vybírá, která stránka se uživateli zobrazí. V MVC aplikaci se zobrazovací komponenta view stará jen o zobrazení dat uživateli a veškeré uživatelské vstupy a iterace zpracovává controller.

4.4 Entity framework

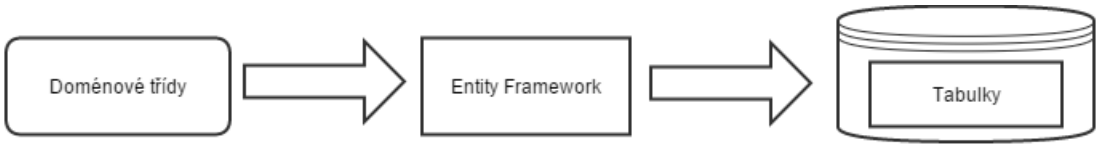
Entity framework poskytuje vývojářům možnost pracovat s relačními daty jako s doménovými objekty, vyřazuje potřebu vytvářet sekce, které museli obvykle psát, aby se dostali k datům. Použitím Entity frameworku se vývojář dotazuje na data pomocí Language Integrated Query (LINQ), tím získá silně typované objekty. Entity framework poskytuje služby jako je sledování, rozlišení identity, dodatečného načtení (lazy loading) a překlad dotazů, takže se vývojáři mohou zaměřit na logiku aplikace místo na potřeby pro získávání dat ze zvolených zdrojů, například z databáze[3].

Entity framework lze použít třemi různými způsoby. První, kdy už máme existující databázi. Druhý, kdy máme vytvořené doménové třídy a databáze se vytvoří podle nich. Třetí možností je vytvořit databázové schéma ve vizuálním designéru a poté teprve vytvořit databázi a třídy. Pro svůj projekt jsem se rozhodl použít druhý způsob, kdy vytvářím modely, ze kterých se generuje databáze. Tento způsob se také nazývá Code first.

4.4.1 Code first

Tato technologie byla představena v Entity frameworku verze 4.1. S tímto přístupem je možné při editaci doménový tříd měnit nebo vytvářet tabulky v databázi. Při vývoji takto navržené aplikace můžeme databázi kdykoliv smazat a znovu vygenerovat. Pokud chceme upravit tabulky, ve kterých jsou již uložená data, lze využít metody migrací[4].

Na obrázku 1.12 je znázorněno, jak si Entity framework vytváří databázi podle přístupu Code first. Pro vývojáře je dostatečné, když vytvoří v programu doménové třídy a pomocí Entity frameworku dokáže vytvořit hotovou databázi.



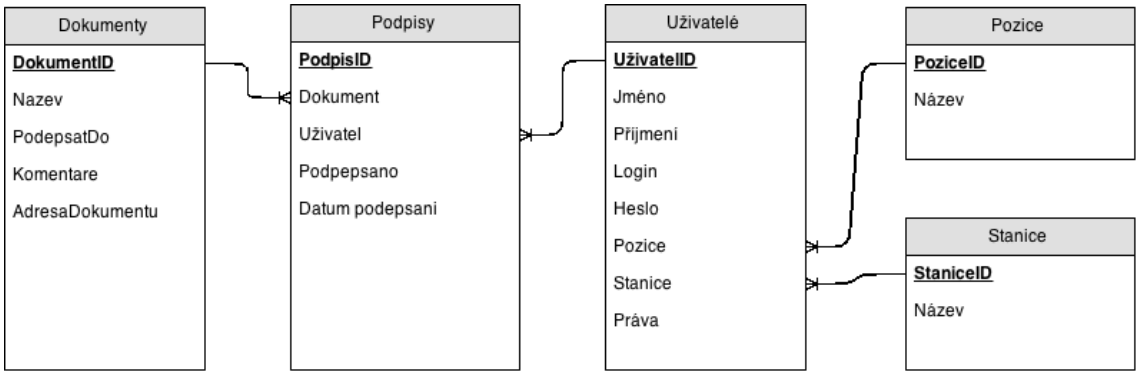
Obrázek 1.12: *Grafické zobrazení Code first přístupu*

4.4.2 Migrace

S pomocí migrací dokážeme upravovat databázové tabulky, aniž bychom je museli celé smazat. Ve vyvíjené aplikaci musíme nejprve migrace povolit. Existují dva druhy migrací: automatické a manuální. Automatické migrace se provedou vždy, když upravíme doménový model. Nevýhodou této metody je to, že nemáme zpětný záznam těchto změn a nemůžeme nahrát všechny změny naráz místo malých změn. Manuální migrace musíme aktivovat příkazy a poskytují nám soubory se záznamy o změnách. Pomocí těchto souborů můžeme zpětně změny vrátit. Migrace si uchovávají snímek původní databáze a poté jen ukládají změny provedené na databázi.

4.5 Databáze

Pro ukládání informací o dokumentech, podpisech a uživatelích je zvolený postup vkládání dat do databáze. Pro tento informační systém je zvolena databáze typu MS SQL od společnosti Microsoft. Vytvoření databáze zajistí rámec Entity framework.



Obrázek 1.13: *Schéma databáze*

Na obrázku 1.13 je zobrazeno schéma databáze, se kterou systém pracuje. V systému jsou zabudovány funkce pro správu dokumentů, podpisů a uživatelů. Pro tabulky pozice a stanice jsou realizovány pouze funkce ke čtení dat z těchto tabulek a očekává se, že data jsou do těchto tabulek nahrána z vnějšího, na tomto systému nezávislého zdroje.

4.6 Bootstrap

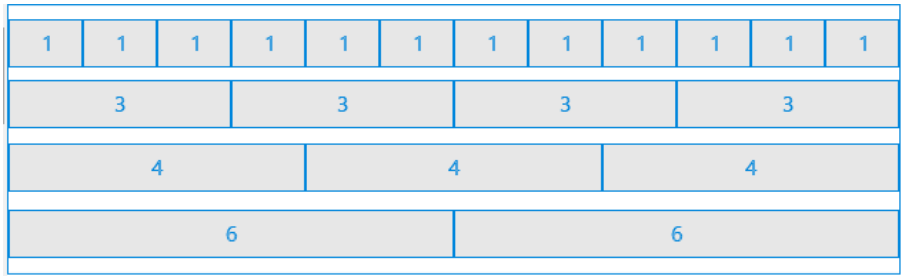
Bootstrap je rámec vytvořený společností Twitter, který je navržen tak, aby podpořil tvorbu webových aplikací a stránek. Poskytuje mnoho užitečných pomůcek, jako základní CSS a HTML pomůcky pro typografii, ikony, formuláře, tlačítka nebo tabulky. Bootstrap má podporu responzivních rozvržení a je testován a podporován na většině hlavních prohlížečích[5].

Obrázek 1.14: *Responzivní rozvržení*

Největší výhodou použití rámce Bootstrap je, že s ním je získání široké škály komponent pro vytváření flexibilních a responzivních webových rozvržení. Navíc, používáním datových metod aplikačního rozhraní (API) můžeme vytvořit pokročilé rozhraní bez nutnosti psát složité JavaScriptové funkce[5].

Mezi výhody použití rámce Bootstrap patří to, že šetří čas, poskytuje responzivní vlastnosti, vytváří konzistentní design a je kompatibilní s dnešními nejznámějšími prohlížeči. Rámec Bootstrap je poskytován zdarma. Díky responzivní kompatibilitě dokážeme stránku navrhnout tak, aby zobrazení vypadalo podle našich představ na různých zařízeních s rozlišnými velikostmi obrazu[5].

Na obrázku 1.15 je zobrazena mřížka bootstrapu, která znázorňuje možnosti rozložení prvků na stránce. Při změně velikosti zobrazení se dokáží jednotlivé prvky přesunout, aby vyhovovaly velikosti displeje a vše bylo dobře čitelné. Ukázka tohoto systému je na obrázku 1.14.

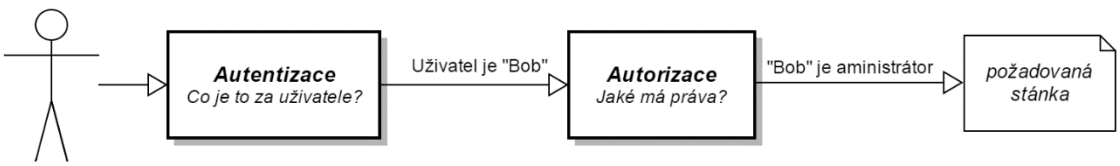


Obrázek 1.15: *Rozvržení Bootstrapové mřížky*

4.7 Bezpečnost, autorizace a autentizace

Pří zabezpečování systému proti neoprávněnému užívání systému musíme uživateli omezit rozsah přístupu v systému. Každý uživatel má své vlastní přihlašovací údaje, které jsou uživateli sděleny při vytvoření daného účtu. Uživatelovy údaje jsou pak uloženy do databáze, kam má přístup jen administrátor a jeho heslo je hashováno.

Na obrázku 1.16 je znázorněn postup procesů z hlediska bezpečnosti, kde je znázorněno, jaké podmínky musí uživatel splnit proto, aby mu byla zpřístupněna požadovaná stránka. Prvním krokem je proces autentizace, kde se ověří přihlašovací údaje a uživatel je jednoznačně identifikován. Dalším krokem je Autorizace, kdy se zjišťuje, zda má uživatel s danými právy přístup k požadované stránce. Pokud splnil obě tyto podmínky, tak je uživateli zobrazena požadovaná stránka.



Obrázek 1.16: *Postup při přístupu uživatele ke stránce*

Autentizace je znalost identity uživatele. Slouží k jednoznačnému určení uživatele, který přistupuje do systému. Systém prohledá databázi, pokud se v ní daný uživatel se správnou kombinací jména a hesla skutečně nachází a pokud ne, tak přihlášení odmítne. Pokud je daná kombinace správná, tak je uživatel přihlášen a dochází k autorizaci.

Autorizace slouží k ověření, zda daný uživatel má právo k provedení nějaké akce. Tento systém je rozdělený na 3 typy účtů s různými právy. Nachází se zde administrátor, který má přístup ke všem prvkům systému. Administrátor je jediný, který může do systému přidávat nebo ubírat uživatele. Celkový počet účtu s administrátorskými právy by měl být omezen. Druhým stupněm oprávnění je manažer, který má přístup ke správě dokumentů, dokáže vytvářet nové dokumenty a přiřazovat je zaměstnancům. Předpokládaný počet uživatelů s těmito právy by neměl přesáhnout několik desítek. Posledním typem je pak účet s právy uživatele, který má přiřazený většina zaměstnanců. Slouží pouze k přístupu do systému, zobrazení patřičných dokumentů a podepisování dokumentů.

Pro zajištění správné autorizace v systémech MVC se používá metoda Authorize. Pomocí komponenty na zjištění rolí v systému se přihlášený uživatel dostane jen na místa, která má povolena. Ovládací prvky jsou zobrazeny jen na stránkách, na které má uživatel přístup. Přístup do specifických sekcí systému pomocí URL je v případě nedostatečného oprávnění zamezen. V takovém případě uživatele systém přesměruje na přihlašovací okno, kde se přihlásí pomocí účtu, který má dostatečná práva pro jeho činnost.

4.8 Tiskové zprávy

Pro vytvoření tiskových zpráv se nabízely dvě reálné možnosti. První možností je vytvoření speciální HTML stránky se svými vlastními CSS styly, které budou poskytovat možnost vytisknout stránku přímo z prohlížeče. Tato možnost má nevýhodu v tom, že existuje mnoho rozličných druhů prohlížečů. Ne každý prohlížeč může danou stranu zobrazit stejným způsobem. Použitím jiného prohlížeče by mohlo dojít k ořezání tiskových sestav a celkově k nevzhlednému formátu vytisknutých dokumentů. Proto je zvolena jiná možnost, a to sice, že systém vygeneruje PDF soubor, který se poté vytiskne. Výhodou je stejný formát tisku u všech použitých prohlížečů. Nevýhodou tohoto způsobu je nutnost stažení PDF souboru přes samotným tiskem. Pro vytvoření PDF tiskových sestav je použita knihovna ITextSharp.

IText je PDF knihovna, která dovoluje vytvářet, upravovat a číst dokumenty ve formátu PDF. Je schopna generovat dokumenty a reporty založené na XML souborech nebo databázích. Dokáže vytvářet mapy a knihy a využívat rozličné funkce souborů formátu PDF[9].

5 Nasazení systému

Posledním neméně důležitým krokem při vývoji informačního systému je jeho nasazení. S komplexnějším systémem je potřeba mít vypracovanou dobrou nasazovací strategii, která bude realizovat nejen prvky daného systému, ale i pomocné prvky, bez nichž by systém nefungoval.

Nasazení systému je inspirováno procesem dle [8].

Pro nasazování systému je dobré mí přístup k různým verzím systému. Většinou se bude používat poslední verze systému, ale při případné nefunkčnosti novější verze je zde možnost vrátit se k původní verzi systému, která ještě fungovala.

Pro nastavování a spravování konfiguračních dat můžeme použít software configuration management (SCM). To zajistí konfigurační údaje, implementuje změněné procesy, verzování dat a uchovávání důležitých údajů. Každá verze má své označení.

Aby bylo nasazování systému co nejvíce uživatelsky přívětivé, je potřeba, aby bylo nasazení systému závislé na uživatelských právech subjektu a ne na jeho znalostech doménového systému. Přestože pro nasazení systému je nutné znát mnoho znalostí o doménovém systému, měly by být tyto informace zpravovány již ve skriptech. Uživatel pak vybere skript, který splňuje jeho potřeby a požadovanou verzi systému.

Ne vždy se však povede nainstalovat systém v pořádku. Proto musí být definovány plány, které se provedou, pokud se při nasazování systému objeví chyba. Nejjednodušší možností, co dělat při chybě vzniklé při nasazení, je vrátit systém do stavu, ve kterém pro daného uživatele fungoval.

Nasazení nového systému

Při nasazení nového systému na úplně nové zařízení se musí zajistit několik důležitých aspektů. Mezi tyto aspekty patří kontrola, zda systém podporuje všechny technologie potřebné pro běh systému. V tomto případě je nutná kontrola, zda systém dokáže pracovat s technologií ASP.NET MVC a MS SQL. Dalším krokem je nahrání systému pomocí skriptu, kde je potřeba jenom určit údaje k připojení do databáze a na webové stránky. Skript poté nahraje systém a vytvoří databázi z implicitními daty pro administrátora systému. Pokud jsou dostupná data ze staršího systému, můžeme získat údaje z minulé databáze a nahrát je do nové. Poslední fázi nasazení je pak jeho kontrola, zda vše funguje tak jak má.

Závěr

Cílem této bakalářské práce bylo navrhnout a vytvořit funkční systém, který firmě poskytne jednoduchou a spolehlivou možnost, jak informovat uživatele o důležitých dokumentech.

V první části práce byl celý systém postupně analyzován z pohledu různých potřeb, které jsou na systém kladeny. Byl zvolen způsob, jakým se bude systém analyzovat, poté byly zodpovězeny základní otázky na požadavky, které musí systém ve výsledku splňovat. Následně jsem pomocí případu užití znázornil základní operace systému. To mi umožnilo konzultovat řešení s vedoucím práce a zástupcem firmy AWT a.s. Díky tomu bylo možno vytvořit nákresy systému, které byly zákazníkovi předběžně ukázány. Poté se návrh zaobírá technickými požadavky, které jsou důležité pro pozdější nasazení systému, aby se případný cílový subjekt rozhodl, s jakou možností se může pracovat.

Druhá polovina práce se pak zabývá samotným vytvořením systému, kde se rozhodne, jakou architekturu bude systém využívat. Pro potřeby této práce byla zvolena třívrstvá architektura, která poskytuje optimální řešení problému.

Z hlediska zadání bakalářské práce se podařilo realizovat všechny potřebné části pro vytvoření fungujícího systému. Jsou vytvořeny internetové stránky pomocí technologie MVC a tyto stránky jsou uvedeny do testovacího provozu. Administrátor zde může spravovat jednotlivé uživatele, kteří budou mít do daného systému přístup. Dále je vytvořena komponenta pro práci s dokumenty, které splňují všechny potřebné požadavky, které byly uvedeny v zadání. Komponenty pro tisk jsou řešené pomocí integrace knihovny ITextSharp do systému. Ta nabízí vytvoření tiskových sestav ve formátu PDF. Posledním problémem bylo realizovat zálohu systému, která je realizována administrátorem, který může předem informovat, že bude provedena záloha.

Celá práce se zabývá jen jednou sekcí, a to práci s dokumenty, ale systém nabízí možnost rozšíření i o další sekce, jako jsou například vnitřní komunikace přes emaily nebo vnitřní zpravodajský oběžník.

V rámci bakalářské práce jsem se seznámil s novými technologiemi. Ač jsem s architekturou spokojen, některé prvky bych mohl navrhnout jinak, a to například systém pro zálohování dat, kdy by se systém mohl zálohovat automaticky. Dalším vylepšením by mohlo být zobrazování PDF dokumentů přímo na stránce bez nutnosti stahovat soubor.

Použitá literatura

[1] JAVED, Muhammad, Bashir AHMAD, Muhammad AHMAD JAN, Muhammad ALI ABID a Muhammad ALI SHAH. RUP Certification via CRM Certification Process: Development of Software with Zero Defect Rate. RUP Certification via CRM Certification Process: Development of Software with Zero Defect Rate [online]. 2012, č. 48, s. 10 [cit. 2015-04-30]. Dostupné z: http://www.sersc.org/journals/IJAST/vol48/1.pdf

[2] ASP.NET MVC Overview. MICROSOFT. MSDN: Developer Network [online]. © 2015 [cit. 2015-04-29]. Dostupné z: https://msdn.microsoft.com/cs-cz/library/dd381412%28v=vs.108%29.aspx

[3] What is Entity Framework?. Entity Framework Tutorial [online]. © 2015 [cit. 2015-04-29]. Dostupné z: http://www.entityframeworktutorial.net/what-is-entityframework.aspx

[4] What is Code-First?:. Entity Framework Tutorial [online]. © 2015 [cit. 2015-04-29]. Dostupné z:http://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx

[5] Bootstrap Introduction. Tutorial republic [online]. © 2015 [cit. 2015-04-29]. Dostupné z:http://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-introduction.php

[6] MIČKA, Pavel. Strategy. Algoritmy.net: příručka vývojáře [online]. © 2008-2015 [cit. 2015-04-29]. Dostupné z: http://www.algoritmy.net/article/1639/Strategy

[7] The Repository Pattern. MICROSOFT. MSDN: Developer Network [online]. © 2015 [cit. 2015-04-29]. Dostupné z: https://msdn.microsoft.com/en-us/library/ff649690.aspx

[8] DOLL, Brian. The rules of software deployment. Emphaticsolutions.com [online]. 2009 [cit. 2015-04-29]. Dostupné z:http://emphaticsolutions.com/2009/09/06/the-rules-of-software-deployment.html

[9] ITextSharp, a .NET PDF library. Sourceforge.net [online]. 31.1.2003, 2.3.2015 [cit. 2015-04-29]. Dostupné z: http://sourceforge.net/projects/itextsharp/

[10] Zachman framework. NIST: Engineering Laboratory [online]. 2011 [cit. 2015-04-30]. Dostupné z: http://www.mel.nist.gov/msid/SSP/standard_landscape/Zachman.html

Seznam obrázků

Seznam příloh

Součástí BP je DVD.

Adresářová struktura přiloženého DVD:

Program

Dokumentace

Skript na vytvoření databáze